

FOUO" 00000000

APPENDIX A

© 1996, 1997, 1998, and 1999 Science Applications International Corporation

BEST AVAILABLE COPY


```

325                                     ;...or Rec Frame Overflow (RFO)
                                     ;...in EXIR
STS_OTHER    BIT STATUS.4 ;General status bit to indicate
                                     ;...an unexpected error type
                                     ;...reported by SCC
330 STS_PCE    BIT STATUS.5 ;Protocol Error reported
                                     ;...on last reception
STS_RAB      BIT STATUS.6 ;Received Aborted message
                                     ;...status from SCC
RES_PKT      BIT STATUS.7 ;Indicates last PKT never sent or aborted - there is
335                                     ;...a slight chance this bit is set in error if POLL from
                                     ;...Master comes after JNB PKT_QUED and before or during
                                     ;...MOV A,#XRES in QUE_PKT sub.
;STS_CRC     BIT STATUS.7 ;Checksum error in last
                                     ;...received message
340 CNTRS_LTCHD BIT PROG_F.0 ;Indicates cntrs latched and ready to be read to buffer.
                                     ;...Set in T0 int and clr'd after cntrs read.
PKT_QUED     BIT PROG_F.1 ;Set after first 32 bytes of a pkt have been loaded into XFIFO.
                                     ;...When set EX0 will set TF1 on next XPR to force T1 int
                                     ;...which finishes pkt transmission once initiated.
345 SYNC_CMND  BIT PROG_F.2 ;Set in Seimens int when PKT_SYNC or TST_PKT_SYNC cmnd rec'd.
                                     ;...Polled during cntr read to buffer to abort and execute cmnd.
DATA_TYPE    BIT PROG_F.3 ;Clr if data type is real counts in PKT mode, set if test counts.
                                     ;...Set in PKT_SYNC or TST_PKT_SYNC cmnd.
PKT_BUF      BIT PROG_F.4 ;Indicates current buffer for new cntr data in PKT mode. If clr
350                                     ;...loading CNTR_BUF0, if set loading CNTR_BUF1.
PKT_IN_PROG  BIT PROG_F.5 ;Set in PKT_SYNC or TST_PKT_SYNC cmnd to indicate PKT mode. All
                                     ;...non-PKT cmnds will call STOP_PKT to clr and end PKT mode.

NEW_CMND     BIT PROG_F.6 ;Set when a valid command
                                     ;...rec'd, cleared when cmnd
355 XFIFO_RDY  BIT PROG_F.7 ;Set on Trans Pool Ready (XPR)
                                     ;...int, cleared when XFIFO
                                     ;...is written to.

360 XSEG                                     ;External Hardware Definitions.
                                     ;...READ/WRITE addresses
PKT_BUF0     EQU 0000H ;Starting address of buffer 0 in on-board 1K RAM.
PKT_BUF1     EQU 0200H ;Starting address of buffer 1 in on-board 1K RAM.
                                     ;...Each buffer is 1/2K (512 bytes) so can handle max PKT size of 16.
365 DIP_SW     EQU 04FFH ;DIP Switch Address. LB must be
                                     ;...FF to avoid bus contention on
                                     ;...external memory read cycle (this may not
                                     ;...be needed w/stretch memory). HB must be
                                     ;...04 so as not to access internal 1K SRAM.

370 CNTR_0     EQU 4000H ;Counter Channel-0
CNTR_1       EQU 4100H ;Counter Channel-1
CNTR_2       EQU 4200H ;Counter Channel-2
CTRL02       EQU 4300H ;Control Word (0-2)
CNTR_3       EQU 4800H ;Counter Channel-3
375 ;000000
CTRL35       EQU 4B00H ;Control Word (3-5)
CNTR_6       EQU 5000H ;Counter Channel-6
CNTR_7       EQU 5100H ;Counter Channel-7
CNTR_8       EQU 5200H ;Counter Channel-8
380 CTRL68     EQU 5300H ;Control Word (6-8)
;000000
CNTR_F       EQU 6800H ;Counter Channel-15
CNTR_10      EQU 6900H ;Counter Channel-16
CNTR_11      EQU 6A00H ;Counter Channel-17
385 CTRLF11    EQU 6B00H ;Control Word (15-17)
LLD_0        EQU 7000H ;8-Bit DAC Discriminator
;...Control (0-15)
; Addresses 8000H to 0BF00H are uncommitted

390 ;-----SAB82526 SCC Register Addresses-----
FIFO         EQU 0C040H ;32 byte Transmit/Receive
                                     ;...FIFO. A write from this
                                     ;...address goes into XFIFO.
                                     ;...a read comes from RFIFO.
395 ISTA       EQU 0C060H ;Interrupt status (read)
MASK         EQU 0C060H ;Interrupt mask (write)
STAR         EQU 0C061H ;Status register (read)
CMR          EQU 0C061H ;Command register (write)
MODE         EQU 0C062H ;Mode register (r/w)
400 TIMR       EQU 0C063H ;Timer register (r/w)
EXTR         EQU 0C064H ;Extended interrupt (read)
XAD1         EQU 0C064H ;Transmit address 1 (write)
RACL         EQU 0C065H ;Receive byte count low (read)
XAD2         EQU 0C065H ;Transmit address 2 (write)

```

```

405 RAH1      EQU 0C066H ;Receive address high 1 (write)
    RSTA     EQU 0C067H ;Receive status reg (read)
    RAH2     EQU 0C067H ;Receive address high 2 (write)
    RAL1     EQU 0C068H ;Receive address low 1 (r/w)
    RHC1     EQU 0C069H ;Receive HDLC control (read)
410 RAL2     EQU 0C069H ;Receive address low 2 (write)
    XBCL     EQU 0C06AH ;Transmit byte count low (write)
    BGR      EQU 0C06BH ;Baudrate generator reg (write)
    CCR2     EQU 0C06CH ;Channel configuration 2 (r/w)
    RBCH     EQU 0C06DH ;Receive byte count high (read)
415 XBCH     EQU 0C06DH ;Transmit byte count high (write)
    VSTR     EQU 0C06EH ;Version status (read)
    RLCR     EQU 0C06EH ;Receive frame length check (write)
    CCR1     EQU 0C06FH ;Channel configuration 1 (r/w)
    TSAX     EQU 0C070H ;Time-slot assignment trans (write)
420 TSAR     EQU 0C071H ;Time-slot assignment rec (write)
    XCCR     EQU 0C072H ;Transmit channel capacity (write)
    RCCR     EQU 0C073H ;Receive channel capacity (write)

;----- Assembler Constants -----
425 ;
;----- SCC CMND Reg Commands -----
RMC      EQU 80H      ;Rec Mess Complete - indicates data
                    ;...has been read following RME int,
                    ;...frees RFIFO space.
430 RHR      EQU 40H      ;Reset HDCL Receiver - clears all data
                    ;...from RFIFO and resets sequence number
                    ;...counters, N(R) and N(S).
    XRES     EQU 01H      ;Transmit Reset - XFIFO is cleared,
                    ;...any mess is aborted and XPR int
                    ;...is generated.
435 XIF      EQU 04H      ;Transmit I Frame - initiates the trans
                    ;...of I frame, address and control field
                    ;...automatically added by SCC. Used when
                    ;...32 bytes written to XFIFO but mess
440          ;...not complete.
    XIF_XME  EQU 06H      ;XIF & Trans Mess End - indicates data in
                    ;...XFIFO completes frame. CRC and closing
                    ;...flag automatically added by SCC.
    XTF      EQU 08H      ;Transmit Transparent Frame - initiates the
445          ;...transmission of Transparent frame, address
                    ;...and control field must be added by uC. Used
                    ;...when 32 bytes written to XFIFO but mess
                    ;...not complete.
    XTF_XME  EQU 0AH      ;XTF & Trans Mess End - indicates data in
450          ;...XFIFO completes frame. CRC and closing
                    ;...flag automatically added by SCC.

    MSTR_ADD EQU 050H      ;Address of Master (System Controller PC)
    SDLC_CTL EQU 003H      ;SDLC Control byte for insertion when sending
455          ;...transparent frames.

    CTL_WD   EQU 30H      ;82c54 Control Word for
                    ;...2-byte I/O, Mode=0, hex
    DAC_INIT EQU 0E1H      ;Power up with optimum DAC (discriminator) setting of 225 dec
460 BROADCAST EQU 0FFH      ;Reserved Broadcast (Global) ATTN address
    RD_BAK   EQU 0CEH      ;Readback command to latch
                    ;...Status and Count of all
                    ;...3 82c54 counters

465 XTAL_MHZ EQU 24        ;Crystal frequency in Mhz. Adjust T0LOAD when changed.

    TMP_DLY  EQU 65535-(1000*XTAL_MHZ) ;Timer 0 value for 12mS
                    ;...delay for DS1260 Init.
                    ;...12mS/[12 (osc. periods/count)] * 1000000
                    ;...((osc. periods/S) = 1000
470 T0LOAD    EQU 256-200    ;Timer 0 reload for 100uS interrupt. (XTAL_MHZx10e6 osc/S /
                    ;...12 osc/cnt) * 100x10e-6 = 200 (for XTAL_MHZ = 24).

;---- System Controller to PCB Commands -----
475 READ_CNTR EQU 00H      ;Stop counting, latch counts, clear and restart counters,
                    ;...read latched counts, load data into XFIFO of SCC
    RD_TEMP  EQU 01H      ;Reads DS1620 temp sensor and loads value into XFIFO.
    RD_DACS  EQU 02H      ;Loads XFIFO with current DAC settings taken from DAC_SAV
                    ;...buffer
480 CLR_CNTRS EQU 03H      ;Stop, clear and restart all counters without reading them.
    RES_XNR  EQU 04H      ;Resets SCC's Receiver and Transmitter and clears diagnostic
                    ;...counters GC_RCV, RL_RCV and NUMERR.
    TEST_DATA EQU 05H      ;Increments each of 16 2-byte dummy count regs and loads
485          ;...into XFIFO along with 2 overflow bytes and STATUS.
                    ;...Overflow bytes set to 0FFH when count regs roll over.

```

```

SND_DEBUG EQU 06H      ;Load diagnostic counters, GC_RCV, RL_RCV and NUMERR into XFIFO.
SET_DACS EQU 07H      ;This command is rec'd with 16 bytes of data. Loads each DAC
                        ;...with corresponding value from receive buffer.
SND_STS EQU 08H        ;Resets the SCC Receiver and Transmitter and loads STATUS
                        ;...into XFIFO. This command is executed internally when any
                        ;...transmit or receive error is detected. Can also be sent
                        ;...by Master.
490 RESEND EQU 09H      ;Reloads CNTR_BUF into XFIFO. CNTR_BUF contains either real
                        ;...cnts from last READ_CNTS cmdnd or dummy cnts from last
                        ;...TEST_DATA cmdnd.
495 PKT_SYNC EQU 00AH   ;
PKT_ASYNC EQU 00BH   ;
TST_PKT_SYNC EQU 00CH ;

500
CSEG
ORG 0000H
LJMP RE_SET          ;Vector to initialization code
505
ORG 0003H
LJMP XINT0_ISR       ;Vector to external INTO ISR. Interrupt from SCC.
ORG 000BH
LJMP TMR0_ISR        ;Vector to TIMER0 ISR.
510
ORG 0013H
LJMP XINT1_ISR       ;Vector to external INT1 ISR. Not used.
ORG 001BH
LJMP TMR1_ISR        ;Vector to TIMER1 ISR. Simulated Int to transmit PKT.
515
ORG 0023H
LJMP SER_ISR         ;Vector to Serial (UART) ISR. Not used.
520
ORG 0100H
USING 1              ;BANK (1) GPRs
;*****
525 ;***** EXTERNAL0_ISR *****
;*****
;Handles int req from Siemens SAB 82526. Sets NEW_CMND if a valid command rec'd.
;Sets XFIFO_RDY if 82526 is ready for more transmit data. Time to service XPR int
;is critical in PKT mode. In PKT mode has second priority to T0, otherwise highest
530 ;priority.
XINT0_ISR:
setb sprpl7
CLR PX0 ;this does not appear to work - see pg 18 of Eng NB - ok to remove
535 PUSH ACC          ;Save Accumulator
PUSH DPL             ;Save Data Pointer Low
PUSH DPH             ;Save Data Pointer High

MOV DPTR,#ISTA       ;SCC Int Status reg
MOVX A,@DPTR         ;Read ISTA
540 JNB ACC.4,CK_RME   ;Jump if not XMIT Pool Ready (XPR)
SETB XFIFO_RDY       ;Indicates XFIFO ready for more data
JNB PKT_QUED,EXIT_ISR_XPR
SETB TFL
CLR PKT_QUED
545 EXIT_ISR_XPR:
POP DPH
POP DPL
POP ACC
SETB PX0
550 clr sprpl7
RETI

CK_RME:
555 PUSH PSW          ;Save Processor Status Word
PUSH R0_RB1
PUSH B
SETB RS0             ;Select BANK(1) GPR's
JNB ACC.7,CK_RPF     ;Jump if not Rec Message End (RME) or XPR
MOV R0,A             ;Save ISTA for error handling if Rec error found
560 MOV DPTR,#RSTA     ;SCC Receive Status Register
MOVX A,@DPTR         ;Check for OK message
ANL A,#0F0H          ;Mask off low nibble of RSTA
CJNE A,#0A0H,REC_ERR ;VFR and CRC ok bits set if good frame
MOV DPTR,#RBCL       ;SCC Rec Byte Count Low reg
565 MOV R0,#REC_CNT_LO ;RAM Storage
MOVX A,@DPTR         ;Read count low byte

```



```

MOV @R0,A          ;...and store
MOV DPTR,#RBCH      ;SCC Rec Byte Count High reg
INC R0              ;Pt to RAM storage
570 MOVX A,@DPTR      ;Read count high byte
MOV @R0,A           ;...and store
CJNE A,#40H,MESS_ERR ;If count HB not 0, too
                    ;...many bytes. NRM bit set
                    ;Max length of valid command
575 MOV A,#19
CLR C               ;Prepare for subb
DEC R0              ;Pt to REC_CNT_LO
SUBB A,@R0          ;Rec counts HB is 0, from above
JC MESS_ERR         ;Exit if command too long
MOV B,@R0           ;Load cntr for bytes rec'd
580 MOV DPTR,#FIFO   ;First address or SCC RFIFO
MOV R0,#HDLN_CNTL  ;Start of our rec buffer
RD_RFIFO_LP:
MOVX A,@DPTR        ;Get byte from RFIFO
MOV @R0,A           ;Load into rec buffer
585 INC R0           ;Count off bytes read
DJNZ B,RD_RFIFO_LP  ;Loop till done
MOV R0,#COMMAND     ;Pt to COMMAND byte of message
MOV A,#0CH          ;Max value of valid command
CLR C               ;Prepare for subb
590 SUBB A,@R0       ;Check COMMAND byte in rec'd mess
JC MESS_ERR         ;...and exit if not valid
SETB NEW_CMND       ;Indicates new valid command rec'd
MOV DPTR,#STAR      ;SCC Status reg
CLEAR:
595 MOVX A,@DPTR
JB ACC.2,CLEAR      ;Wait til Command Executing (CEC) is clear
MOV DPTR,#CMDR      ;SCC Command reg
MOV A,#RMC          ;Send Rec Message Complete command
MOVX @DPTR,A        ;...to release space in RFIFO
600 CJNE @R0,#PKT_SYNC,CK_TST_SYNC
SETB SYNC_CMND
CK_TST_SYNC:
CJNE @R0,#TST_PKT_SYNC,NOT_SYNC
SETB SYNC_CMND
605 NOT_SYNC:
SJMPC EXIT_ISR      ;Done with message reception. Valid command
                    ;...rec'd with no errors.
MESS_ERR:
610 SETB STS_INVALID ;Indicate a valid frame rec'd but message was invalid
SJMPC STS_SET
REC_ERR:
JB ACC.7,CK_RDO     ;RSTA in ACC
SETB STS_OTHER      ;Jump if Valid Frame Ready (VFR)
615 CK_RDO:
JNB ACC.6,CK_CRC    ;Indicate Receive Data Overflow error
SETB STS_RDO_RFO
CK_CRC:
JB ACC.5,CK_RAB     ;Indicate a checksum error
620 ; SETB STS_CRC    ;Set Other bit for unexpected ints
SETB STS_OTHER
CK_RAB:
JNB ACC.4,NO_ABORT  ;Jump if Rec Aborted Mes (RAB) not set
SETB STS_RAB        ;Indicate RAB error
625 NO_ABORT:
MOV A,R0            ;Restore ISTA to ACC following REC_ERR
CK_RPF:
JNB ACC.6,CK_EXIR   ;Jump if not Rec Pool Full (RPF)
SETB STS_OTHER      ;Set Other bit for unexpected ints
630 CK_EXIR:
JNB ACC.0,STS_SET    ;If no other intrpts, finish error handling
MOV DPTR,#EXIR      ;Extended Int Reg. - check other sources
MOVX A,@DPTR        ;...of interrupt
JNB ACC.7,CK_XDU     ;Jump if not XMIT Message Repeat (XMR)
635 SETB STS_OTHER
CK_XDU:
JNB ACC.6,CK_PCE     ;Indicate Transmit Data Underrun (XDU)
SETB STS_XDU
CK_PCE:
640 JNB ACC.5,CK_RFO  ;Jump if not Protocol Error (PCE)
SETB STS_PCE        ;Indicate PCE in Status byte but don't
CK_RF:
JNB ACC.4,STS_SET    ;Indicate Rec Frame Overflow (RFO)
SETB STS_RDO_RFO
645 STS_SET:
MOV COMMAND,#8      ;If error, execute send status command
;..... this instr does not work! indirect addressable only

```

```

        SETB NEW_CMND          ;Indicates new valid command.
EXIT_ISR:
650      POP B
        POP R0_RB1
        POP PSW              ;Restore PSW
        POP DPH              ;Restore DPH
        POP DPL              ;Restore DPL
655      POP ACC              ;Restore ACC
        SETB PX0
        clr sprp17
        RETI

660      ;#####
        ;#                TIMER0 ISR                #
        ;#####
        ;Used as highest priority interrupt in packet mode. Started with PKT_SYNC or
665      ;TST_PKT_SYNC commands to dec software timer (LTCH_CNTR_TMR) and latch counters
        ;when 0. Set up as 8-bit auto-reload to reduce overhead. This mode will work
        ;for timer resolution of up to about 125 uS. For larger values of TOLoad 16 bit
        ;timer mode must be used. Min time to service is essential for fastest comm rate.
        ;Must be able to interrupt T1 and return before XFIFO is emptied.
670      ;On power-up timer is set to model and used by INIT_DS1260 to provide 12mS delay.
        TMR0_ISR:
        setb sprp15
        PUSH PSW              ;Save processor status
        SETB RS0              ;Select BANK(1) GPRs
675      INC R6
        CJNE R6,#0FFH,EXIT_T0_ISR
        CJNE R7,#0FFH,INC_MSB
        SJMP LATCH_CNTRS
        INC_MSB:
        INC R7
680      clr sprp15
        SJMP EXIT_T0_ISR
        LATCH_CNTRS:          ;Same as READ_CNTRS subroutine through CALL RESET_CNTRS instr.
        setb sprp16
685      PUSH ACC
        PUSH CKCON
        PUSH DPL
        PUSH DPH
        ORL CKCON,#001H      ;Set MD0 bit for stretch memory of 1.
690      MOV DPTR,#CTRL02      ;Point DPTR at 82c54(0,3)
        MOV R2,#6             ;Readback all 6 82c54 ICs
        CLR GATE              ;Disable counting
        LOOP10:
        MOV A,#RD_BAK         ;Readback command to latch Status and Count of all 3
695      MOVX @DPTR,A          ;...counters of the IC
        MOV A,DPH              ;Send readback to IC(1)
        MOV A,DPH              ;ACC=DPH
        ADD A,#8              ;Add offset to next 82c54
        MOV DPH,A             ;Update DPH
700      DJNZ R2,LOOP10
        ; MOV DPTR,#CNTR_0      ;Point DPTR at 82c54(0)
        ; MOV R2,#6             ;Preload all 6 82c54 ICs
        ;LOOP09:
        ; MOV A,#0FFH
705      ; MOVX @DPTR,A          ;Preload CNTR_0 LSB
        ; MOVX @DPTR,A          ;Preload CNTR_0 MSB
        ; INC DPH
        ; MOVX @DPTR,A          ;Preload CNTR_1 LSB
        ; MOVX @DPTR,A          ;Preload CNTR_1 MSB
710      ; INC DPH
        ; MOVX @DPTR,A          ;Preload CNTR_2 LSB
        ; MOVX @DPTR,A          ;Preload CNTR_2 MSB
        ; MOV A,DPH              ;ACC=DPH
        ; ADD A,#6              ;Add offset to next 82c54
715      ; MOV DPH,A             ;Update DPH
        ; DJNZ R2,LOOP09
        SETB GATE              ;Re-enable counting
        DJNZ R5,PKT_NOT_DUN
        MOV R1,#0B9H
720      MOV CUR_CNTR_LTCH_TM_LB,@R1
        INC R1
        MOV CUR_CNTR_LTCH_TM_HB,@R1
        PKT_NOT_DUN:
        MOV R6,CUR_CNTR_LTCH_TM_LB
725      MOV R7,CUR_CNTR_LTCH_TM_HB
        SETB CNTRS_LTCHD
        POP DPH
        POP DPL

```

```

730     POP CKCON
       POP ACC
       clr sprp16
       EXIT_TO_ISR:
       POP PSW
       RETI
735

;#####
;#                               TIMER1 ISR                               #
;#####
740 ;Simulated interrupt. Initiated in EX0 when an XPR Int occurs indicating the start
;of a PKT transmission. Has lowest priority. Loads 32 bytes from buffer @DPTR1 into
;XFIFO then loops until XFIFO_RDY is set. Continues until buffer is empty. Speed is
;critical to allow a T0 Int to occur (possibly with a latch cntrs requirement) and
;not have XFIFO empty.
745 TMR1_ISR:
       setb sprp17
       PUSH ACC
       PUSH CKCON
       PUSH DPL
750       PUSH DPH
       PUSH PSW           ;Save Processor Status Word
       PUSH P2
       SETB RS0           ;Select BANK(1) GPR's
       ANL CKCON,#0FEH    ;Set to no stretch memory.
755       MOV DPL,PKT_BUF_OUT_LB
       MOV DPH,PKT_BUF_OUT_HB
       MOV R4,#32
       MOV P2,#0C0H
       MOV R0,#040H
760       CLR XFIFO_RDY
       LOAD_XFIFO:
       MOVX A,@DPTR
       MOVX @R0,A
       INC DPTR
765       DJNZ R4,LOAD_XFIFO
       MOV R0,#061H
       MOV A,#XTF
       MOVX @R0,A
       WT_FOR_XPR:
770       JNB XFIFO_RDY,WT_FOR_XPR
       CLR XFIFO_RDY
       MOV R0,#040H
       MOV R4,#32
       DJNZ R3,LOAD_XFIFO
775       MOV R4,#3
       LD_MSG_END:
       MOVX A,@DPTR
       MOVX @R0,A
       INC DPTR
780       DJNZ R4,LD_MSG_END
       MOV R0,#061H
       MOV A,#XTF_XME
       MOVX @R0,A
       MOV R0,#NEW_PKT_SIZ
785       MOV A,@R0
       CLR C
       SUBB A,#2
       MOV SND_PKT_SIZ,A
       POP P2
790       POP PSW
       POP DPH
       POP DPL
       POP CKCON
       POP ACC
795       clr sprp17
       RETI

       USING 0                               ;BANK (0) GPRs
800 ;#####
;#                               RESET AND ENABLE ALL COUNTERS                               #
;#####
; Disable all 16 counters by clearing GATE, then
;preload all counters to FFFFH (which also resets
805 ;the OUT and NULL Status Flags), and lastly re-
;enable counters by resetting the GATE. It is
;possible that a counter may accumulate one count
;(but no more than one count) between preloading
;of its MSB and resetting of the GATE. Call with stretch

```

```

810 ;memory of 1.
    RESET_CNTRS:
        CLR GATE ;Disable counting
        MOV DPTR,#CNTR_0 ;Point DPTR at 82c54(0)
        MOV R2,#6 ;Preload all 6 82c54 ICs
815 LOOP02:
        MOV A,#0FFH
        MOVX @DPTR,A ;Preload CNTR_0 LSB
        MOVX @DPTR,A ;Preload CNTR_0 MSB
        INC DPH
820 MOVX @DPTR,A ;Preload CNTR_1 LSB
        MOVX @DPTR,A ;Preload CNTR_1 MSB
        INC DPH
        MOVX @DPTR,A ;Preload CNTR_2 LSB
        MOVX @DPTR,A ;Preload CNTR_2 MSB
825 MOV A,DPH ;ACC=DPH
        ADD A,#6 ;Add offset to next 82c54
        MOV DPH,A ;Update DPH
        DJNZ R2,LOOP02
        SETB GATE ;Re-enable counting
830 RET
;
;
;*****
;# LATCH ALL COUNTERS AND QUE DATA, #
835 ;*****
; Disable all 16 counters by clearing GATE, then
; issue the readback command to all 6 82c54s (which
; latches the Status and Count). Next, call the
; RESET_CNTRS subroutine to reset and re-enable all
840 ;counters and then que the data into the counter
;buffer. Also set the corresponding overflow bit
;of any counters that overflow. <- not yet implemented *****
;
READ_CNTRS:
845 ORL CKCON,#001H ;Set MD0 bit for stretch memory of 1.
;...Counters require a stretch of 1 even
;...at 16 MHz to satisfy worst case
;...conditions. Even with stretch can't
;...go much above 24 MHz for uC crystal.
;...This instruction assumes MD1 and MD2
;...are clear.
850 MOV DPTR,#CTRL02 ;Point DPTR at 82c54(0,3)
        MOV R2,#6 ;Readback all 6 82c54 ICs
        CLR GATE ;Disable counting
855 LOOP03:
        MOV A,#RD_BAK ;Readback command to latch
;...Status and Count of all 3
;...counters of the IC
;Send readback to IC(i)
860 MOVX @DPTR,A
        MOV A,DPH ;ACC=DPH
        ADD A,#8 ;Add offset to next 82c54
        MOV DPH,A ;Update DPH
        DJNZ R2,LOOP03
        CALL RESET_CNTRS ;Reset and re-enable all 16
865 ;...counters
        MOV R3,#0 ;Init IC counter, cntr (3 counters/IC)
        MOV R0,#CNTR_BUF ;Point to counter buffer
        MOV DPTR,#CNTR_0 ;Point DPTR at 1st IC, 1st counter
LOOP04:
870 MOVX A,@DPTR ;Read Status of IC(i,j)
        JNB ACC.7,NO_OVF ;Jump if no overflow
        SETB CTROVF ;Set the cntr overflow flag
;...of the STATUS byte
NO_OVF:
875 MOV C,ACC.6 ;Save NULL bit of Status
        MOVX A,@DPTR ;Read count LSB of IC(i,j)
        MOV B,A ;Save LSB at B
        MOVX A,@DPTR ;Read count MSB of IC(i,j)
        JNC NO_NUL ;Jump if counter has been
880 ;...triggered at least once
        MOV B,#0 ;Clear the count LSB
        CLR A ;Clear the count MSB
NO_NUL:
885 MOV R5,A ;R5=MSB
        CLR A ;Subtract contents of down-
        CLR C ;...counter from 10000H
        SUBB A,B ;LSB of difference
        MOV @R0,A ;Save LSB of diff in cntr buf
        INC R0 ;Pointer to data MSB
890 CLR A

```

```

SUBB A,R5          ;MSB of difference
MOV @R0,A          ;Save MSB of diff in cntr buf
INC R0             ;Point to next counter LSB
INC DPH            ;Point to next counter
995 CJNE R0,#OVRFLW0_7,MORE_CNTRS
    SJMP READ16     ;Jump at end of counter buffer.
                    ;...2 cntrs not used

MORE_CNTRS:
    CJNE R0,#ODD_BEGIN,MORE_CNT2 ;When pntr reaches here,
900                                ;...indicates we are at last counter of
                                ;...even bank (U14,CNTR2) which is not used.
    INC DPH            ;Skip unused counter.
    SJMP NXT_IC       ;Go To Next IC
MORE_CNT2:
905 INC R3
    CJNE R3,#3,LOOP04 ;Loop for all 3 counters of
                    ;...each 82c54
NXT_IC:
    MOV R3,#0         ;Reset for next IC (3 counters/IC)
910 MOV A,DPH          ;Mov DPTR to next IC, 1st counter
    ADD A,#5
    MOV DPH,A
    SJMP LOOP04       ;Get next 3 counters
READ16:
915 ANL CKCON,#0FEH   ;Set to no stretch memory.
    RET

;#####
920 ;# SET DACS DIFF #
;#####
;Writes values from rec buffer to corresponding DAC. Correct order
;determined by Master. Also copies new DAC values to DAC_SAV buffer.
;DACs require stretch memory of 1.
925 SET_DACS_DIFF:
    ORL CKCON,#001H   ;Set MD0 bit for stretch memory of 1.
                    ;...DACs require a stretch of 1 to
                    ;...satisfy worst case conditions.
                    ;...This instruction assumes MD1 and MD2
                    ;...are clear.
930 MOV DPTR,#LLD_0    ;Point DPTR at DAC(0)
    MOV R0,#DACV       ;Point R0 to DAC val location
    MOV R1,#DAC_SAV
LOOP16:
935 MOV A,@R0          ;Move Desired DAC val in R0
    MOVX @DPTR,A       ;Write value to DAC(m)
    MOV @R1,A          ;Write value to DAC_SAV(m)
    INC DPH            ;Point DPTR at next DAC
    INC R0             ;Next new value to set to
940 INC R1             ;Point to next save spot
    CJNE R0,#DACV+16,LOOP16 ;Loop for 16 DACs
    ANL CKCON,#0FEH   ;Set to no stretch memory.
    RET

945 ;#####
;# SET ALL DACS #
;#####
;Call with desired DAC value in ACC which will
950 ;then be written to all 16 DACs. Also copies new DAC values
;to DAC_SAV buffer. DACs require stretch memory of 1.
;
SET_ALL_DACS:
    ORL CKCON,#001H   ;Set MD0 bit for stretch memory of 1.
                    ;...DACs require a stretch of 1 to
                    ;...satisfy worst case conditions.
                    ;...This instruction assumes MD1 and MD2
                    ;...are clear.
955 MOV DPTR,#LLD_0    ;Point DPTR at DAC(0)
    MOV R0,#DAC_SAV   ;Point R0 at DAC_SAV(0)
LOOP06:
    MOVX @DPTR,A       ;Write value to DAC(m)
    MOV @R0,A          ;Write valude to DAC_SAV(m)
    INC DPH            ;Point DPTR at next DAC
965 INC R0
    CJNE R0,#DAC_SAV+16,LOOP06 ;Loop for 16 DACs
    ANL CKCON,#0FEH   ;Set to no stretch memory.
    RET

970 ;#####

```

```

;# READ_DAC_SETTINGS
;*****
;Loads current DAC settings, which are stored in DAC_SAV buffer,
;into XFIFO.
975 READ_DAC_SETTINGS:
    MOV R0, #DAC_SAV
    MOV B, #16
    CALL XMIT_DATA
980    RET

;***** DS1620 (DIGITAL THERMOMETER) DRIVERS *****
;#
985 ;***** INITIALIZE DS1620 *****
; Call during powerup reset to configure the DS1620 for continuous sampling
;operation with a CPU and initiate the first temperature conversion. Uses R2 of
;the currently selected Register Bank. Uses Timer 0 so it should be undedicated and
990 ;it's interrupt should be disabled.
INIT_DS1620:
    SETB SCK ; Make sure the clock is high
    SETB SEL ; Chip Select DS1620
    MOV A, #0CH ; Write CONFIG Command
995    CALL WRT_8BR
    MOV A, #02H ; CONFIG Byte- (Continuous mode with CPU)
    CALL WRT_8BR
    CLR SEL ; De-select the DS1620
    CLR TR0 ; Ensure Timer 0 is off
1000    CLR TFO ; Ensure Timer 0 OF flag is reset
    MOV TLO, #LOW(TMP_DLY) ; Reload value for a
    MOV TH0, #HIGH(TMP_DLY) ; ...12mS delay
    SETB TR0 ; turn on Timer 0, interrupt should be disabled
WAIT_12mS: ; Allow time for DS1620 EEPROM write cycle
1005    JNB TFO, WAIT_12mS ; Jump til Timer 0 overflows
    CLR TR0 ; Leave with Timer 0 off
;----- Start 1st Temp Conversion
    SETB SEL ; Chip Select DS1620
    MOV A, #0EEH ; Start Convert T Command (initiate first Temp.
1010    CALL WRT_8BR ; ...conversion - data ready in 1 s)
    CLR SEL ; De-select the DS1620
    RET

;***** READ DS1620 TEMPERATURE *****
1015 ; Read DS1620 and return the Centigrade temperature in TEMP_MSB, TEMP_LSB,
;with 0.5 °C resolution and in two's complement format. That is, a return
;value of FF92H = -55 °C and a return value of 00FAH = +125 °C. The DS1620
;returns a 9 bit 2's complement # in the range of -55 °C to +125 °C, which is
;converted to a full 2 byte 2's complement number.
1020 READ_TEMP:
    SETB SCK ; Make sure the clock is high
    SETB SEL ; Chip Select DS1620
    MOV A, #0AAH ; Read TEMP Command
    CALL WRT_8BR
1025    SETB DIO ; Make sure DIO is Hi-Z input
    MOV R2, #8
    LOOP004:
    CLR SCK ; Clock data from DS1620 onto DIO
    MOV C, DIO ; Pickup data-bit from DIO
1030    RRC A ; Rotate data-bit into ACC
    SETB SCK ; Set serial clock high
    DJNZ R2, LOOP004 ; Loop for 8 LSB's
    CLR SCK ; Clock 9th bit from DS1620 onto DIO
    MOV C, DIO ; Pickup data-bit from DIO
1035    SETB SCK ; Set serial clock high
    CLR SEL ; De-select the DS1620
    MOV TEMP_LSB, A
    MOV TEMP_MSB, #00 ; Assume temp >= 0
    JNC MSB_DONE
1040    MOV TEMP_MSB, #0FFH ; Create full 2 byte, 2's complement #
    MSB_DONE:
    RET

;***** 8-BIT DS1620 WRITE ROUTINE *****
1045 ; Call with data byte in ACC. Each ACC-bit is written (LSb first) onto DIO
;and followed by a 0/1 transition of SCK. Original data is trashed. Uses R2 of
;the currently selected Register Bank.
WRT_8BR:
    MOV R2, #8
1050    LOOP002:
    CLR SCK ; Set serial clock low
    RRC A ; Shift next LSb into CARRY

```

```

MOV DIO,C          ; Move data onto DIO
SETB SCK           ; Clock data into serial device
1055 DJNZ R2,LOOP002 ; Write all 8 bits of ACC
RET

;***** TRANSMIT DATA *****
1060 ; Load data into XFIFO of SCC and execute Transmit Transparent Frame command.
;Call with R0 pointing to first byte of data and B containing number of
;bytes to transmit, excluding STATUS which will always be loaded as first
;byte of every message. If XFIFO is not clear, it will be reset prior to
;loading data. Therefore any existing data that has not been retrieved by
1065 ;Master will be lost. If more than 32 bytes are to be sent, program
;execution will loop in this routine until transmission has been initiated
;by the Master, allowing remaining bytes to be loaded. The Address and Control
;fields of the SDLC message format are written to XFIFO as required when trans-
;mitting transparent frames. Uses R1.
1070 XMIT_DATA:
JB XFIFO_RDY,LOAD_DATA ;Skip reset FIFO if XFIFO ready
MOV DPTR,#CMDR         ;SCC Command reg (w), Status reg (r)
FINISH_CMND:
MOVX A,@DPTR
1075 JB ACC.2,FINISH_CMND ;Check CEC bit and jump if a cmdnd executing
CLR PKT_QUED          ;Be sure XPR from XRES does not trigger T1 Int
MOV A,#XRES           ;Load reset XFIFO command
MOVX @DPTR,A
WT_FOR_INT:
1080 JNB XFIFO_RDY,WT_FOR_INT ;Wait for Trans Pool Ready INT
;...after XFIFO reset
LOAD_DATA:
CLR XFIFO_RDY         ;Indicate XFIFO not ready during trans
MOV A,#MSTR_ADD       ;Masters address must be inserted for transparent
MOV DPTR,#FIFO        ;...frame.
1085 MOVX @DPTR,A        ;Load into XFIFO
MOV A,#SDLC_CTL       ;SDLC Control byte must be inserted for transparent
MOVX @DPTR,A         ;...frame.
MOV A,STATUS          ;STATUS is 1st byte of every trans
MOVX @DPTR,A
1090 MOV STATUS,#0      ;Current status sent, so clear
MOV A,B               ;Load send count into ACC
CLR C                 ;Prepare for subtraction
SUBB A,#29            ;Check if send count > 29
MOV R1,#0             ;Assume its not & clear 2nd block counter
1095 JC LOOP_B         ;If not > 29 then jump to send "B" bytes
MOV B,#29             ;If > 29 then 1st block = 29 plus Adrs,Cntrl & STATUS
MOV R1,A              ;Load 2nd block counter with remainder
LOOP_B:
MOV A,#R0              ;R0 initialized to first byte to send
1100 MOVX @DPTR,A        ;Load byte in XFIFO
INC R0                ;Pt to next byte
DJNZ B,LOOP_B         ;Load "B" bytes
JC MSG_END            ;If C set from SUBB, then entire message loaded
MOV DPTR,#STAR        ;SCC Status reg
1105 FINISH_CMND1:
MOVX A,@DPTR
JB ACC.2,FINISH_CMND1 ;Check CEC bit and jump if a cmdnd executing
MOV DPTR,#CMDR        ;Send Transmit Transparent Frame command
MOV A,#XTF            ;...to SCC to send 32 byte
1110 MOVX @DPTR,A        ;...block.
WT_FOR_INT1:
JNB XFIFO_RDY,WT_FOR_INT1 ;Wait for Trans Pool Ready INT
CLR XFIFO_RDY         ;Indicate XFIFO not ready during transmission
MOV DPTR,#FIFO        ;Pt to XFIFO
1115 FINISH_DATA:
MOV A,#R0              ;Load data byte
MOVX @DPTR,A          ;...into XFIFO
INC R0                ;Pt to next byte
DJNZ R1,FINISH_DATA   ;Send remainder of message
1120 MOV DPTR,#STAR     ;SCC Status reg
FINISH_CMND2:
MOVX A,@DPTR
JB ACC.2,FINISH_CMND2 ;Check CEC bit and jump if a cmdnd executing
MSG_END:
1125 MOV DPTR,#CMDR     ;Send XTF and Trans Message End
MOV A,#XTF_XME        ;...command to SCC to finish
MOVX @DPTR,A          ;...Transparent frame.
RET
1130
;*****
; QUE PKT FOR XMISSION AND RESET PKT BUFFER
;*****

```

```

;Initiate transmission of current cntr buffer (PKT) and set up the next buffer for
1135 ;new data. Get here after CNTR_RDS has been dec'd to 0 in T0 int.
      QUE_PKT:
        MOV P2,#0COH
        JNB PKT_QUED,OLD_PKT_XMITTD
        MOV R0,#061H ;SCC Command reg (w), Status reg (r)
        MOV A,#XRES ;Load reset XFIFO command
1140 CLR PKT_QUED
        MOVX @R0,A ;Master must handle aborted frame
        SETB RES_PKT ;Indicates last PKT never sent or aborted - there is
                        ;...a slight chance this bit is set in error if POLL from
                        ;...Master comes after JNB PKT_QUED and before or during
                        ;...MOV A,#XRES.
1145
      OLD_PKT_XMITTD:
        MOV R0,#NEW_PKT_SIZ ;Pt to PKT size in rec'd message
        MOV CNTR_RDS,@R0 ;Set up PKT cntr for current PKT
1150 JNB PKT_BUF_LD_BUF1
        MOV PKT_BUF_IN_LB,#LOW(PKT_BUF0)
        MOV PKT_BUF_IN_HB,#HIGH(PKT_BUF0)
        JMP SET_OUTBUF
      LD_BUF1:
1155 MOV PKT_BUF_IN_LB,#LOW(PKT_BUF1)
        MOV PKT_BUF_IN_HB,#HIGH(PKT_BUF1)
      SET_OUTBUF:
        CPL PKT_BUF
        JNB PKT_BUF_SND_BUF1
1160 MOV DPL,#LOW(PKT_BUF0)
        MOV DPH,#HIGH(PKT_BUF0)
        JMP PKT_QUE
      SND_BUF1:
        MOV DPL,#LOW(PKT_BUF1)
1165 MOV DPH,#HIGH(PKT_BUF1)
      PKT_QUE:
        JB XFIFO_RDY,LOAD_PKT ;Skip reset FIFO if XFIFO ready
        MOV R0,#061H ;SCC Command reg (w), Status reg (r)
        MOV A,#XRES ;Load reset XFIFO command
1170 MOVX @R0,A
      WT_FOR_INT3:
        JNB XFIFO_RDY,WT_FOR_INT3 ;Wait for Trans Pool Ready INT
        ;...after XFIFO reset
      LOAD_PKT:
        CLR XFIFO_RDY ;Indicate XFIFO not ready during trans
1175 MOV A,#MSTR_ADD ;Masters address must be inserted for transparent
        MOV R0,#040H ;...frame.
        MOVX @R0,A ;Load into XFIFO
        MOV A,#SDLC_CTL ;SDLC Control byte must be inserted for transparent
        MOVX @R0,A ;...frame.
1180 MOV A,STATUS ;STATUS is 1st byte of every trans
        MOVX @R0,A
        MOV STATUS,#0 ;Current status sent, so clear
        MOV B,#29
      LOOP_B1:
1185 MOVX A,@DPTR
        MOVX @R0,A
        INC DPTR
        DJNZ B,LOOP_B1 ;Load "B" bytes
        MOV R0,#061H ;SCC Command reg (w), Status reg (r)
1190 MOV A,#XTF
        MOVX @R0,A
      WT_FOR_INT4:
        JNB XFIFO_RDY,WT_FOR_INT4 ;Wait for Trans Pool Ready INT
        ;...after XFIFO reset
1195 CLR XFIFO_RDY
        MOV R0,#040H
        MOV B,#32
      LOOP_B2:
        MOVX A,@DPTR
        MOVX @R0,A
1200 INC DPTR
        DJNZ B,LOOP_B2 ;Load "B" bytes
        MOV R0,#061H ;SCC Command reg (w), Status reg (r)
        MOV A,#XTF
        MOV PKT_BUF_OUT_LB,DPL
1205 MOV PKT_BUF_OUT_HB,DPH
        SETB PKT_QUED
        MOVX @R0,A
        ;Must be here in case XPR occurs due to master poll
        ;This XTF should not cause an XPR int because both XFIFOs are
        ;...now full. If an XPR occurs immediately then a poll from master
        ;...must have occurred after first XTF above. Since PKT_QUED and
        ;...PKT_BUF_OUT already handled then vector to T1 int is ok.
        ;...Normally after returning from QUE_PKT, MAIN loop and T0 int run
1210
      ong
        ;...until master polls and XPR (w/PKT_QUED set) causes T1 int to tr
        ;...entire PKT.

```



```

1215 RET

;***** READ CNTR LATCHES TO PKT BUFFER *****
;*****
1220 ;Read the latched cntrs into PKT_BUF_IN
RD_CNTRS_PKT:
    MOV DPL,PKT_BUF_IN_LB
    MOV DPH,PKT_BUF_IN_HB
    MOV P2,#HIGH(CNTR_0)
1225    MOV R0,#LOW(CNTR_0) ;Point R0/P2 at 1st IC, 1st counter
    MOV R1,#CNTR_BUF ;Point to storage of last count data
    MOV R3,#0 ;Init IC counter, cntr (3 counters/IC)
    ORL CKCON,#001H ;Set MD0 bit for stretch memory of 1.
LOOP01:
1230    JB SYNC_CMND,READALL
    MOVX A,@R0 ;Read Status of IC(i,j)
    MOVX A,@R0 ;Read count LSB of IC(i,j)
    XCH A,@R1
    CLR C
1235    SUBB A,@R1
    INC R1
    MOVX @DPTR,A ;Save LSB of diff in cntr buf
    INC DPTR
    MOVX A,@R0 ;Read count MSB of IC(i,j)
1240    XCH A,@R1
    SUBB A,@R1
    INC R1
    MOVX @DPTR,A ;Save LSB of diff in cntr buf
    INC DPTR ;Pointer to data MSB
1245    INC P2 ;Point to next counter
    MOV A,P2
    CJNE A,#HIGH(CNTR_11),MORE_CNTRS1
    SJMP READALL ;Jump at end of counter buffer.
;...2 cntrs not used
1250 MORE_CNTRS1:
    CJNE A,#HIGH(CNTR_8),MORE_CNTRS2 ;When pntr reaches here,
;...indicates we are at last counter of
;...even bank (U14,CNTR2) which is not used.
    INC P2 ;Skip unused counter.
1255    SJMP NXT_IC1 ;Go To Next IC
MORE_CNTRS2:
    INC R3
    CJNE R3,#3,LOOP01 ;Loop for all 3 counters of
;...each 82c54
1260 NXT_IC1:
    MOV R3,#0 ;Reset for next IC (3 counters/IC)
    MOV A,P2 ;Mov P2 to next IC, 1st counter
    ADD A,#5
    MOV P2,A
1265    SJMP LOOP01 ;Get next 3 counters
READALL:
    ANL CKCON,#0FEH ;Set to no stretch memory.
    MOV PKT_BUF_IN_LB,DPL
    MOV PKT_BUF_IN_HB,DPH
1270 RET

;***** INC TST DATA AND READ TO PKT BUFFER *****
;*****
1275 ;*****
TDATA_READ:
    MOV R0,#DUMYO_LSB ;Pt to LB of 1st test data
INC_DATA_LP1:
    INC @R0 ;Increment test data
1280    CJNE @R0,#0,NO_ROLL_OVR1 ;Jump unless byte rolled over
    INC R0 ;If LB rolled, pt to HB
    INC @R0 ;...and increment
    SJMP NEXT_LB1
NO_ROLL_OVR1:
1285    INC R0 ;Pt to data word HB
NEXT_LB1:
    INC R0 ;Pt to next data word LB
    CJNE R0,#DUMYO_7,INC_DATA_LP1 ;Jump if not at end
    MOV R0,#DUMYO_LSB ;Pt to LB of 1st test data word
1290    MOV DPL,PKT_BUF_IN_LB
    MOV DPH,PKT_BUF_IN_HB
LD_TST_DTA:
    MOV A,@R0
    MOVX @DPTR,A

```

```

1295     INC R0
        INC DPTR
        CJNE R0,#DUMYOFO_7,LD_TST_DTA ;Jmp if not at end
        MOV PKT_BUF_IN_LB,DPL
        MOV PKT_BUF_IN_HB,DPH
1300     RET

;#####
;#                               END PACKET MODE
;#####
1305     STOP_PKT:
        CLR TR0
        CLR PKT_IN_PROG
        CLR CNTRS_LTCHD
1310     CLR PKT_QUED
        RET

;#####
;#                               INITIALIZATION ROUTINES
;#####
1315     ; One-Time Initialization Routines.
        RE_SET:
        MOV PMR,#01000101B ;IC=XTAL/4, ALE disabled for onboard
                                ;...data access, use 1K of onboard XRAM
1320     MOV IE,#00001011B ;Enable External INT 0,Timer 0 and Timer 1 (EA cleared).
        MOV IP,#00000011B ;EX0 and T0 have high PRIORITY
        MOV P1,#00011101B ;Hold 82526 in reset condition
                                ;GATE=0
1325     MOV P3,#11111111B ;Set other port pins high
        MOV P2,#11111111B ;...to recreate powerup
        MOV P0,#11111111B ;...configuration
        MOV WDCON,#10000000B ;SMOD1=1 for dbl Baud rate. Not used
                                ;...with UART1
1330     MOV TCON,#00000000B ;Level triggered IRQs if
                                ;...enabled. Also stops
                                ;...both timers and clears
                                ;...both IRQ edge flags
        MOV TMOD,#00000001B ;TMR0 is 16-bit general purpose and TMR1 is Mode 0. TMR0 will
                                ;...be set during PKT SYNC command. TMR1 does not matter since it
1335     MOV CKCON,#00000000B ;Watchdog Timer = 2^17 clocks, TM0,1,2
                                ;...is used as a simulated Int, initiated in EX0.
                                ;...use 12 clocks, no stretch memory
        MOV SP,#STACK-1 ;Initialize Stack Pointer to STACK-1 since SP
                                ;is inc'd before use.
1340     ;----- Clear 87C520 Scratchpad RAM -----
        MOV R0,#0FFH ;Point to end of scratchpad RAM
        LOOP07:
1345     MOV @R0,#0 ;Initialize 87C520 RAM to 0
        DJNZ R0,LOOP07
        CLR RESET_526 ;Take 82526 out of Reset state (1.8uS min)
1350     ;----- Read PC Board Address -----
        ORL CKCON,#001H ;Set MD0 bit for stretch memory of 1. Appears to be marginal
                                ;...timing at full speed as LB of DIP_SW must be FF to avoid
                                ;...bus contention.
1355     MOV DPTR,#DIP_SW ;Point DPTR at DIP Switch
        MOVX A,@DPTR ;Read DIP Switch
        MOV BD_ADR,A ;Save DIP Switch setting
        ANL CKCON,#0FEH ;Set to no stretch memory.
;----- Initialize Temp Sensor -----
1360     LCALL INIT_DS1620 ;Set up sensor and begin conversions. Returns with T0 disabled,
                                ;...TFO clr.
        MOV TMOD,#02H ;T0 set for 8 bit, auto-reload for use with PKT mode
;----- Initialize DACs to Mid-Range -----
        MOV A,#DAC_INIT ;Mid-range setting
1365     CALL SET_ALL_DACs
;----- Configure 82c54 Event Counters -----
        ORL CKCON,#001H ;Set MD0 bit for stretch memory of 1. Counters require
                                ;...a stretch of 1 even at 16 MHz to satisfy worst case
                                ;...conditions. Even with stretch can't go much above
                                ;...24 MHz for uC crystal. This instruction assumes MD1
                                ;...and MD2 are clear.
1370     MOV DPTR,#CTRL02 ;Point DPTR at 0th 82c54
        MOV R0,#6 ;Configure 6 82c54 ICs
        LOOP08:
1375     MOV A,#CTL_WD ;Ctrl Word for counter-0

```

```

MOVX @DPTR,A      ;... (2-byte, Mode=0, hex)
ADD A,#40H         ;Write CW for counter-0
MOVX @DPTR,A      ;Offset to next cntr CW
ADD A,#40H         ;Write CW for counter-1
MOVX @DPTR,A      ;Offset to next cntr CW
1380 MOV A,DPH       ;Write CW for counter-2
      ADD A,#8      ;ACC=DPH
      MOV DPH,A     ;Add offset to next 82c54
1385 DJNZ R0,LOOP08 ;Update DPH
      ANL CKCON,#0FEH ;Set to no stretch memory.

;----- Initialize SAB82526 SCC -----
MOV DPTR,#CCR1    ;Channel Config Reg -
1390 MOV A,#91H    ;Power Up, Bus Configuration
MOVX @DPTR,A      ;...Clock Mode 1
MOV DPTR,#MODE    ;Mode Reg. Auto, 8 bit address
MOV A,#08H        ;...external timer, RTS auto
MOVX @DPTR,A      ;...control, timer res. k=32.768
1395 MOV DPTR,#TIMR ;Timer Reg. CNT(retries)=1
MOV A,#3FH        ;...VALUE=63: t=k*(VALUE+1)*TCP
MOVX @DPTR,A      ;...t=timeout, TCP=clock period
MOV DPTR,#XAD1    ;Transmit Address 1
MOV A,#MSTR_ADD   ;Masters address
1400 MOVX @DPTR,A
MOV DPTR,#XAD2    ;Transmit Address 2
MOV A,#MSTR_ADD   ;Masters address
MOVX @DPTR,A
MOV DPTR,#RAL1    ;Receive Address Low 1
1405 MOV A,BD_ADR  ;Board Address as determined by
MOVX @DPTR,A      ;...switch
MOV DPTR,#RAL2    ;Receive Address Low 2
MOV A,#BROADCAST  ;Used for broadcast address in NRM
MOVX @DPTR,A
1410 MOV DPTR,#RAH1 ;Receive Address High 1
MOV A,#0          ;0 for single byte address
MOVX @DPTR,A
MOV DPTR,#RAH2    ;Receive Address High 2
MOV A,#0          ;0 for single byte address,
1415 MOVX @DPTR,A  ;...modulo 8
MOV DPTR,#XBCH    ;Transmit Byte Count High
MOV A,#40H        ;Interrupt mode, NRM
MOVX @DPTR,A
MOV DPTR,#RLCR    ;Receive Length Check sets max
1420 MOV A,#0      ;...receive length, after which
MOVX @DPTR,A      ;...reception is suspended-disabled
MOV DPTR,#MASK    ;MASK Interrupt disable register
MOV A,#2EH        ;...ICA & EXA channel A interrupts
MOVX @DPTR,A      ;...disabled, RSC, TIN disbl'd
1425 MOV DPTR,#CCR2 ;Channel Control Register 2 RTS
MOV A,#00H        ;...active during trans, TxCLK is..
MOVX @DPTR,A      ;...input, CTS disabled, RFS int disbl'd
MOV DPTR,#CMDR    ;Command Register
MOV A,#00H        ;clear commands
1430 MOVX @DPTR,A
MOV DPTR,#XBCL    ;Transmit Byte Count DMA only
MOVX @DPTR,A      ;clear reg
MOV DPTR,#BGR     ;Baudrate Generator not used
MOVX @DPTR,A      ;clear reg
1435 MOV DPTR,#TSAX ;Time-Slot Assignment clk 5 only
MOVX @DPTR,A      ;clear reg
MOV DPTR,#TSAR    ;Time-Slot Assitnment clk 5 only
MOVX @DPTR,A      ;clear reg
MOV DPTR,#XCCR    ;Transmit Channel Capacity clk 5 only
1440 MOVX @DPTR,A  ;clear reg
MOV DPTR,#RCCR    ;Receive Channel Capacity clk 5 only
MOVX @DPTR,A      ;clear reg

;----- Enable 87CS1 IRQs -----
1445 SETB EA       ;Global IRQ enable

;----- Power Up 82526 -----
MOV DPTR,#STAR    ;SCC Status/Command reg
CLEAR0:
1450 MOVX A,@DPTR
      JB ACC.2,CLEAR0 ;Wait til Command Executing (CEC) is clear

      MOV DPTR,#CMDR ;SCC Command reg
      MOV A,#XRES   ;reset XFIFO
1455 MOVX @DPTR,A
      WT_FOR_INT2:  ;Wait for Trans Pool Ready INT

```

```

JNB XFIFO_RDY,WT_FOR_INT2 ;...after XFIFO reset

1460 MAIN:
    JB NEW_CMND,DO_CMND ;Jump to execute command if pending
    JNB CNTRS_LTCHD,MAIN ;Loop until a new command or counters latched (PKT mode)

    CLR CNTRS_LTCHD ;Reset
    JNB DATA_TYPE,LD_TST_CNTR ;DATA_TYPE set indicates real data (PKT mode)
1465 CALL RD_CNTRS_PKT ;Read cntrs into current PKT_BUF_IN
    SJMP CNTRS_READ
LD_TST_CNTR:
    CALL TDATA_READ ;Inc test data and read into CNTR_BUF_IN
1470 CNTRS_READ:
    JB SYNC_CMND,MAIN ;Poll for PKT_SYNC or TST_PKT_SYNC cmds
    MOV A,CNTR_RDS ;Load current PKT cntr
    JNZ MAIN ;Jump if current PKT still in progress
    CALL QUE_PKT ;Que filled cntr buffer for xmission, point PKT_BUF_IN to
1475 ;...to other buffer
    SJMP MAIN ;Wait for new cmd or new cntr data latched

DO_CMND:
    MOV R0,#COMMAND ;Point to COMMAND byte of rec'd mess
1480 MOV A,R0 ;Get new command from rec buffer

    CJNE A,#PKT_SYNC,CHK_TPKT_SYNC
    CLR TRO
    MOV TLO,#TLOAD ;Count for 100uS Int
1485 MOV TH0,#TLOAD ;Reload for TLO
    SETB TRO
    CLR NEW_CMND
    ORL CKCON,#001H ;Set MD0 bit for stretch memory of 1.
    CALL RESET_CNTRS
1490 ANL CKCON,#0FEH ;Set to no stretch memory.
    MOV R0,#0B9H
    MOV CUR_CNTR_LTCH_TM_LB,R0
    INC R0
    MOV CUR_CNTR_LTCH_TM_HB,R0
1495 MOV LTCH_CNTR_TMR_LB,CUR_CNTR_LTCH_TM_LB
    MOV LTCH_CNTR_TMR_HB,CUR_CNTR_LTCH_TM_HB
    MOV R0,#CNTR_BUF
    MOV A,#0FFH
INIT_LST_CNT_BUF:
1500 MOV R0,A
    INC R0
    CJNE R0,#OVRFLW0_7,INIT_LST_CNT_BUF
    MOV R0,#NEW_PKT_SIZ ;Pt to PKT size in rec'd message
    MOV CNTR_RDS,R0 ;Set up PKT cntr for current PKT
1505 MOV A,R0
    CLR C
    SUBB A,#2
    MOV SND_PKT_SIZ,A
    CLR PKT_BUF
1510 MOV PKT_BUF_IN_LB,#LOW(PKT_BUF0)
    MOV PKT_BUF_IN_HB,#HIGH(PKT_BUF0)
    SETB DATA_TYPE
    SETB PKT_IN_PROG
    CLR CNTRS_LTCHD
1515 CLR SYNC_CMND
    CLR NEW_CMND
    SJMP MAIN

CHK_TPKT_SYNC:
1520 CJNE A,#TST_PKT_SYNC,CHK_PKT_ASYNC
    CLR TRO
    MOV TLO,#TLOAD ;Count for 100uS Int
    MOV TH0,#TLOAD ;Reload for TLO
    SETB TRO
1525 MOV R0,#DUMY0_LSB ;Pt to LB of 1st test data
CLR_TST_DATA:
    MOV R0,#0
    INC R0
    CJNE R0,#DUMY0F0_7,CLR_TST_DATA
1530 MOV R0,#0B9H
    MOV CUR_CNTR_LTCH_TM_LB,R0
    INC R0
    MOV CUR_CNTR_LTCH_TM_HB,R0
    MOV LTCH_CNTR_TMR_LB,CUR_CNTR_LTCH_TM_LB
    MOV LTCH_CNTR_TMR_HB,CUR_CNTR_LTCH_TM_HB
1535 MOV R0,#NEW_PKT_SIZ ;Pt to PKT size in rec'd message
    MOV CNTR_RDS,R0 ;Set up PKT cntr for current PKT

```

```

MOV A, @R0
CLR C
1540 SUBB A, #2
MOV SND_PKT_SIZ, A
CLR PKT_BUF
MOV PKT_BUF_IN_LB, #LOW(PKT_BUF0)
MOV PKT_BUF_IN_HB, #HIGH(PKT_BUF0)
1545 CLR DATA_TYPE
SETB PKT_IN_PROG
CLR CNTRS_LTCHD
CLR SYNC_CMND
CLR NEW_CMND
1550 JMP MAIN

CHK_PKT_ASYNC:
CJNE A, #PKT_ASYNC, CHK_READ_CNTR
CLR NEW_CMND
1555 JMP MAIN

CHK_READ_CNTR:
CJNE A, #READ_CNTR, CHK_RESEND
; READ_CNTR is the primary command that will be used under normal operating
; conditions. It will be sent as a global command to all boards. Stop
; counting, latch count, clear and restart counters, read counter latch and
; load data into XFIFO.
CLR NEW_CMND ; Processing command
; The following code up to NO_ROLL_OVR4 label is for diagnostics only. The Get
; Counts Rec'd counter is transmitted to Master after receipt of SND_DEBUG cmdnd.
1565 MOV R0, #GC_RCV_LSB ; Pt to LSB of # Get Counts Received
INC @R0 ; Increment # Get Counts Received
CJNE @R0, #0, NO_ROLL_OVR4 ; Jump Unless Byte Rolled over
INC R0 ; If LB rolled, pt to next byte
1570 INC @R0 ; ...and increment
CJNE @R0, #0, NO_ROLL_OVR4 ; Jump Unless Byte rolled over
INC R0 ; Pt to next byte
INC @R0 ; ...and increment
CJNE @R0, #0, NO_ROLL_OVR4 ; Jump Unless High byte rolled over
1575 MOV @R0, #0FFH ; Set All #Get Counts bytes to FF
DEC R0
MOV @R0, #0FFH
DEC R0
MOV @R0, #0FFH

1580 NO_ROLL_OVR4:
CALL READ_CNTR
MOV R0, #CNTR_BUF ; Pt to 1st byte in counter buffer
MOV B, #34 ; Number of bytes to XMIT, excluding STATUS
CALL XMIT_DATA ; Trans B bytes beginning at R0
1585 JNB PKT_IN_PROG, NO_PKT
CALL STOP_PKT
NO_PKT:
JMP MAIN

1590 ;
CHK_RESEND:
CJNE A, #RESEND, CHK_TEST_DATA
; RESEND reloads the XFIFO with the last counter data which is stored in RAM
; at CNTR_BUF. The counters are not latched, reset or read.
1595 CLR NEW_CMND ; Processing command
; The following code up to NO_ROLL_OVR2 label is for diagnostics only. The Reloads
; Rec'd counter is transmitted to Master after receipt of SND_DEBUG cmdnd.
MOV R0, #RL_RCV_LSB ; Pt to LSB of # Reloads Received
INC @R0 ; Increment # Reloads Received
1600 CJNE @R0, #0, NO_ROLL_OVR2 ; Jump Unless Byte Rolled over
INC R0 ; If LB rolled, pt to Hb
INC @R0 ; ...and increment
CJNE @R0, #0, NO_ROLL_OVR2 ; Jump Unless HB rolled over
1605 MOV @R0, #0FFH ; Set Both #reload bytes to FF
MOV R0, #RL_RCV_LSB
MOV @R0, #0FFH

NO_ROLL_OVR2:
MOV R0, #CNTR_BUF ; Pt to 1st byte in counter buffer
MOV B, #34 ; Number of bytes to XMIT, excluding STATUS
1610 CALL XMIT_DATA ; Trans B bytes beginning at R0
JNB PKT_IN_PROG, NO_PKT0
CALL STOP_PKT
NO_PKT0:
JMP MAIN

1615 ;
CHK_TEST_DATA:
CJNE A, #TEST_DATA, CHK_SND_STS

```

```

1620 ;TEST DATA Command increments each of 16 2-byte dummy count regs
;and sets 2 overflow bytes to 0FFH on rollover. These 34 bytes plus
;STATUS are then loaded into XFIFO and transmitted.
CLR NEW_CMND ;Processing command
;The following code up to NO_ROLL_OVR5 label is for diagnostics only. The Get
;Counts Rec'd counter is transmitted to Master after receipt of SND_DEBUG cmd.
1625 MOV R0,#GC_RCV_LSB ;Pt to LSB of # Get Counts Received
INC R0 ;Increment # Get Counts Received
CJNE R0,#0,NO_ROLL_OVR5 ;Jump Unless Byte Rolled over
INC R0 ;If LB rolled, pt to next byte
INC R0 ;...and increment
CJNE R0,#0,NO_ROLL_OVR5 ;Jump Unless Byte rolled over
INC R0 ;Pt to next byte
INC R0 ;...and increment
CJNE R0,#0,NO_ROLL_OVR5 ;Jump Unless High byte rolled over
MOV R0,#0FFH ;Set All #Get Counts bytes to FF
1635 DEC R0
MOV R0,#0FFH
DEC R0
MOV R0,#0FFH
NO_ROLL_OVR5: ;Pt to LB of 1st test data
MOV R0,#DUMYO_LSB ;Increment test data
1640 INC_DATA_LP: ;Jump unless byte rolled over
INC R0 ;If LB rolled, pt to HB
CJNE R0,#0,NO_ROLL_OVR ;...and increment
INC R0 ;Jump unless HB rolled over
INC R0 ;Pt to first Overflow byte
CJNE R0,#0,NEXT_LB ;...and set all bits to indicate
MOV R1,#DUMYO0_7 ;...all data words rolled. Pt to
MOV R1,#0FFH ;...2nd OF byte and do same.
INC R1 ;Jump to pt to next data word
1650 MOV R1,0FFH
SJMP NEXT_LB ;Pt to data word HB
NO_ROLL_OVR:
INC R0 ;Pt to next data word LB
NEXT_LB: ;Jump if not at end
1655 INC R0 ;Pt to LB of 1st test data word
CJNE R0,#DUMYO0_7,INC_DATA_LP ;Number of bytes to XMIT, excluding STATUS
MOV R0,#DUMYO_LSB ;Trans B bytes beginning at R0
MOV B,#34 ;Set up Buffer to copy from
CALL XMIT_DATA ;Set up Buffer to copy to
1660 MOV R0,#DUMYO_LSB
MOV R1,#CNTR_BUF
COPY_BUF:
MOV A,R0 ;Copy Buffer byte
MOV R1,A ;Point Source buffer to next byte
INC R0 ;Point Dest buffer to next byte
INC R1 ;Jump if not at end
CJNE R0,#DUMYO0_7,COPY_BUF
JNB PKT_IN_PROG,NO_PKT1
CALL STOP_PKT
1670 NO_PKT1:
JMP MAIN
;
CHK_SND_STS:
1675 CJNE A,#SND_STS,CHK_RD_TEMP
;This Command is set by SCC ISR when any transmit or receive error is detected.
;It can also be sent by Master.
CLR NEW_CMND ;Processing current command
;The following code up to NO_ROLL_OVR3 label is for diagnostics only. The Number
1680 ;of Errors counter is transmitted to Master after receipt of SND_DEBUG cmd.
MOV R0,#NUMERR_LSB ;Pt to LSB of # Errors Detected
INC R0 ;Increment # Errors Detected
CJNE R0,#0,NO_ROLL_OVR3 ;Jump Unless Byte Rolled over
INC R0 ;If LB rolled, pt to Hb
INC R0 ;...and increment
CJNE R0,#0,NO_ROLL_OVR3 ;Jump Unless HB rolled over
MOV R0,#0FFH ;Set Both #error bytes to FF
MOV R0,#NUMERR_LSB
MOV R0,#0FFH
1690 NO_ROLL_OVR3:
MOV DPTR,#CMDR ;SCC Command reg - reset RFIFO
CLEAR1:
MOVX A,@DPTR ;Wait til Command Executing (CEC) is clear
JB ACC.2,CLEAR1 ;Send Reset HDLC Receiver (RHR)
MOV A,#RHR
MOVX @DPTR,A
MOV DPTR,#CMDR
CLR XFIFO_RDY
CLEAR2:

```

```

1700      MOVX A, @DPTR
        JB  ACC.2, CLEAR2          ;Wait til Command Executing (CEC) is clear
        CLR  PKT_QUED              ;Be sure XPR from XRES does not trigger T1 Int
        MOV  A, #XRES              ;Send Transmit Reset (XRES) to reset XFIFO
        MOVX @DPTR, A

1705      WT_FOR_INT5:
        JNB  XFIFO_RDY, WT_FOR_INT5 ;Wait for Trans Pool Ready INT
        ;...after XFIFO reset

        LOAD_STS:
        CLR  XFIFO_RDY            ;Indicate XFIFO not ready during trans
        MOV  A, #MSTR_ADD          ;Masters address must be inserted for transparent
        ;...frame.
1710      MOV  DPTR, #FIFO
        MOVX @DPTR, A              ;Load into XFIFO
        MOV  A, #SDLC_CTL          ;SDLC Control byte must be inserted for transparent
        MOVX @DPTR, A              ;....frame.
        MOV  A, STATUS              ;STATUS is 1st byte of every trans
1715      MOVX @DPTR, A
        MOV  STATUS, #0
        MOV  DPTR, #CMDR
        ;

        CLEAR3:
        MOVX A, @DPTR
1720      JB  ACC.2, CLEAR3          ;Wait til Command Executing (CEC) is clear
        MOV  A, #XTF_XME          ;Send XTF and Trans Message End
        MOVX @DPTR, A              ;...command to SCC to finish I frame.
        JNB  PKT_IN_PROG, NO_PKT2
        CALL STOP_PKT

1725      NO_PKT2:
        JMP  MAIN
        ;

        CHK_RD_TEMP:
1730      ;Reads DS1620 temp sensor and loads value into XFIFO.
        CJNE A, #RD_TEMP, CHK_RES_XNR
        CLR  NEW_CMND              ;Processing current command
        CALL READ_TEMP             ;Read DS1620 into TEMP_LSB/MSB
        JB  XFIFO_RDY, LOAD_TEMP
1735      MOV  DPTR, #CMDR
        ;SCC Command reg - reset XFIFO

        CLEAR4:
        MOVX A, @DPTR
        JB  ACC.2, CLEAR4          ;Wait til Command Executing (CEC) is clear
        CLR  PKT_QUED              ;Be sure XPR from XRES does not trigger T1 Int
1740      MOV  A, #XRES
        MOVX @DPTR, A
        ;

        WT_FOR_INT6:
        JNB  XFIFO_RDY, WT_FOR_INT6 ;Wait for Trans Pool Ready INT
        ;...after XFIFO reset

        LOAD_TEMP:
1745      CLR  XFIFO_RDY            ;Indicate XFIFO not ready during trans
        MOV  A, #MSTR_ADD          ;Masters address must be inserted for transparent
        MOV  DPTR, #FIFO
        MOVX @DPTR, A              ;...frame.
        MOV  A, #SDLC_CTL          ;Load into XFIFO
        MOVX @DPTR, A              ;SDLC Control byte must be inserted for transparent
        MOV  A, STATUS              ;....frame.
1750      MOVX @DPTR, A
        MOV  A, STATUS              ;STATUS is 1st byte of every trans
        MOVX @DPTR, A
        MOV  STATUS, #0
        MOV  A, TEMP_LSB
1755      INC  DPTR
        MOVX @DPTR, A
        MOV  A, TEMP_MSB
        MOVX @DPTR, A
        MOV  DPTR, #CMDR
        ;Send XTF and Trans Message End
1760      MOV  A, #XTF_XME          ;...command to SCC to finish
        MOVX @DPTR, A              ;...I frame.
        JNB  PKT_IN_PROG, NO_PKT3
        CALL STOP_PKT

        NO_PKT3:
1765      JMP  MAIN
        ;

        CHK_RES_XNR:
        ;Resets SCC's Receiver and Transmitter and clears diagnostic counters GC_RCV,
1770      ;RL_RCV and NUMERR.
        CJNE A, #RES_XNR, CHK_SET_DACS
        CLR  NEW_CMND              ;Processing current command
        MOV  DPTR, #CMDR          ;SCC Command reg - reset XFIFO

        CLEAR5:
1775      MOVX A, @DPTR
        JB  ACC.2, CLEAR5          ;Wait til Command Executing (CEC) is clear
        CLR  PKT_QUED              ;Be sure XPR from XRES does not trigger T1 Int
        MOV  A, #XRES              ;...if not ready
        MOVX @DPTR, A

1780      CLEAR6:

```

```

MOVX A, @DPTR
JB ACC.2, CLEAR6
MOV A, #RHR
MOVX @DPTR, A
MOV R0, #GC_RCV_LSB
MOV @R0, #0
1785
NEXTBYTE:
INC R0
MOV @R0, #0
1790 CJNE R0, #NUMERR_MSB, NEXTBYTE
JNB PKT_IN_PROG, NO_PKT4
CALL STOP_PKT
NO_PKT4:
JMP MAIN
1795 ;
CHK_SET_DACS:
;This command is rec'd with 16 bytes of data. Loads each DAC with corresponding value
;from receive buffer.
1800 CJNE A, #SET_DACS, CHK_RD_DACS
CLR NEW_CMND
CALL SET_DACS_DIFF
JNB PKT_IN_PROG, NO_PKT5
CALL STOP_PKT
1805 NO_PKT5:
JMP MAIN
;
CHK_RD_DACS:
;Loads XFIFO with current DAC settings taken from DAC_SAV buffer
1810 CJNE A, #RD_DACS, CHK_CLR_CNTRS
CLR NEW_CMND
CALL READ_DAC_SETTINGS
JNB PKT_IN_PROG, NO_PKT6
CALL STOP_PKT
1815 NO_PKT6:
JMP MAIN
;
CHK_CLR_CNTRS:
;Stop, clear and restart all counters without reading them.
1820 CJNE A, #CLR_CNTRS, CHK_DEBUG
CLR NEW_CMND
ORL CKCON, #001H
1825 ;Set MD0 bit for stretch memory of 1.
;...DACs require a stretch of 1 to
;...satisfy worst case conditions.
;...This instruction assumes MD1 and MD2
;...are clear.
CALL RESET_CNTRS
ANL CKCON, #0FEH
1830 JNB PKT_IN_PROG, NO_PKT7
CALL STOP_PKT
NO_PKT7:
JMP MAIN
1835 ;
CHK_DEBUG:
;This command used for diagnostics only. It transmits the Get Counts Rec'd,
;Reloads Rec'd and Number of Errors counters.
1840 CJNE A, #SND_DEBUG, NO_VAL_CMND
CLR NEW_CMND
MOV R0, #GC_RCV_LSB
MOV B, #7
CALL XMIT_DATA
JNB PKT_IN_PROG, NO_PKT8
CALL STOP_PKT
1845 NO_PKT8:
JMP MAIN
;
1850 NO_VAL_CMND:
JMP MAIN
;
1855 END ; End of VACISBBD.ASM module

```